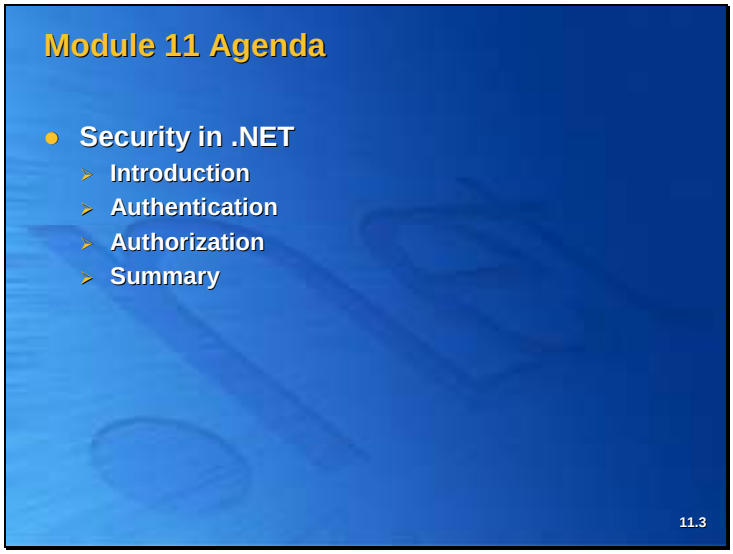
Microsoft
.net developer
training

Security in .NET

Agenda: Day 2

- Module 09: Creating XML Web Services
- Module 10: Code Instrumentation and Middle Tier Technology
- ▶ Module 11: Security in .NET
- Module 12: Lifecycle Tools for .NET
- Module 13: Configuring and Deploying .NET Applications
- Module 14: Application Interoperability
- Module 15: Migrating Applications to .NET
- Module 16: Building Applications for Mobile Devices

 ©2002 Microsoft Corporation



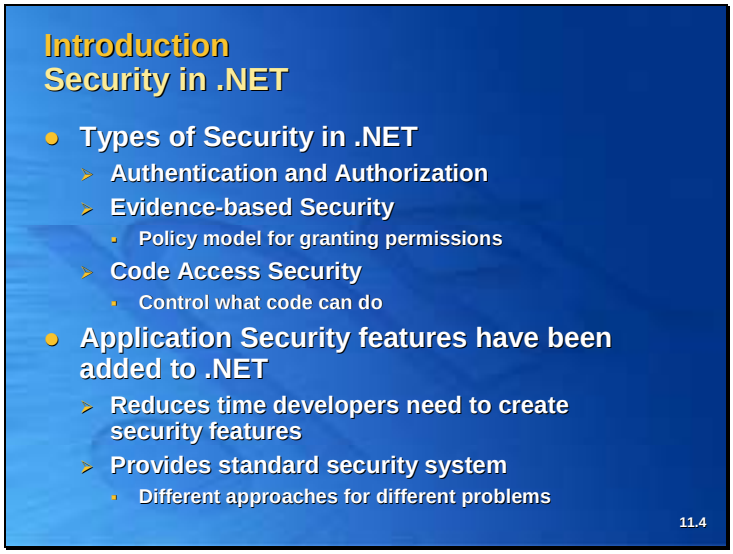
Module 11 Agenda

What you will learn:

The types of security used in .NET. This module focuses on:

- The types of Authentication used in .NET
- The Authorization strategies you can use in .NET

See the Appendix for information on Advanced Security The appendix contains topics for more information on other security topics including Code-based Security and Role-based Security.


Introduction
Security in .NET
Types of Security in .NET
Authentication and Authorization
Evidence-based Security
Policy model for granting permissions
Code Access Security
Control what code can do
Application Security features have been added to .NET
Reduces time developers need to create security features
Provides standard security system
Different approaches for different problems
11.4

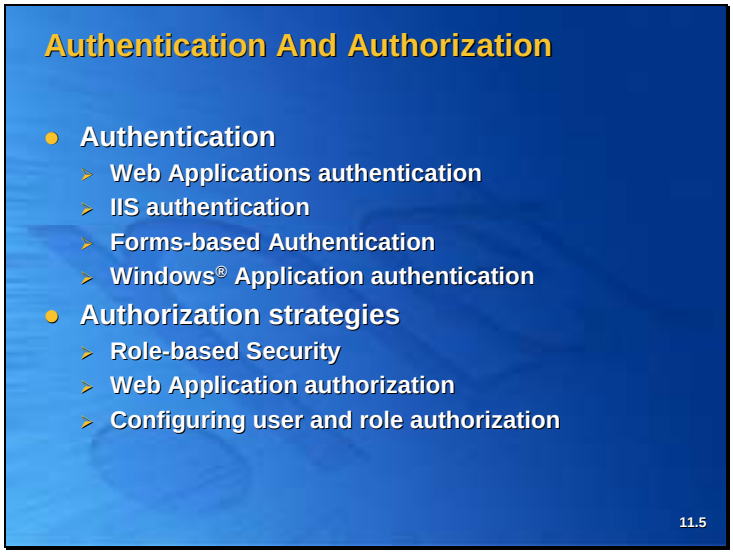
Introduction

Security in .NET

This module discusses new features in .NET Security Authentication and Authorization for Web and Windows Applications to secure in-transit data and improvements in the familiar standards.

.NET Role-based Security is about users, which is the major topic for this module. However, additional security measures afforded by .NET such as Evidence-based Security (evidence-based is about code), Code Access Security, Crypto Security (crypto classes) and other security models are discussed in the following Advanced Security Features presentation.

 ©2002 Microsoft Corporation

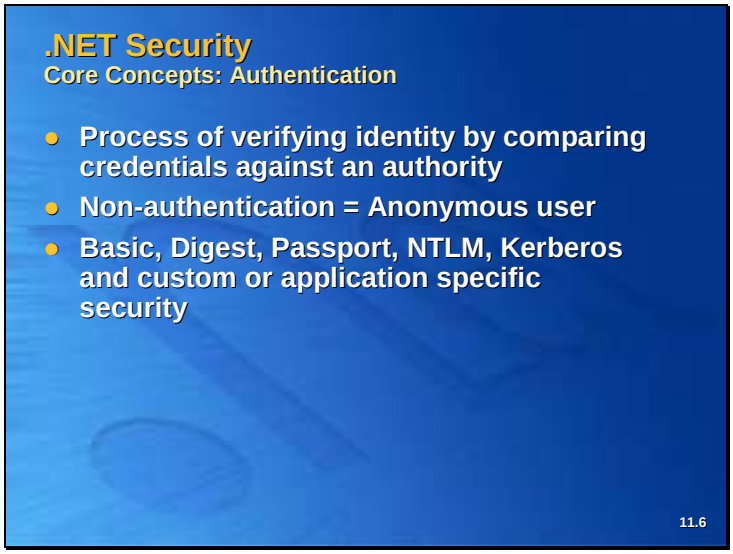


Authentication And Authorization

Authentication is the process of accepting credentials from a user and validating those credentials against some authority. If the credentials are valid, one speaks of having an authenticated identity. Authorization is the process of determining whether that authenticated identity has access to a given resource. Authentication can be accomplished by either system or business logic, and is available through a single API. The authentication API is fully extensible, so developers can use their own business logic as needed. Developers may code their authentication needs to this single API, and may revise the underlying authentication methods without making major changes in their code. In addition to Microsoft Windows® operating system identity authentication, available authentication methods include basic HTTP, Digest, and Kerberos, as well as Microsoft Passport and forms-based authentication. These methods of authentication are also fully integrated into ASP.NET.

In ASP.NET forms authentication, the user provides credentials and submits the forms. If the application authenticates the request, the system issues a cookie that contains the credentials in some form or a key for re-acquiring the identity. Subsequent requests are issued with the cookie in the request headers and they are authenticated and authorized by an ASP.NET handler by using whatever validation method the application desires. If a request is not authenticated, HTTP client-side redirection is used to send that request to an authentication form, where the user can supply authentication credentials. Forms authentication is sometimes used for personalization—the customization of content for a known user. In some of these cases, identification is the issue rather than authentication, so a user's personalization information can be obtained simply by accessing the user name.

The purpose of **authorization** is to determine whether a requesting identity is granted access to a given resource. ASP.NET offers two types of authorization services: file authorization and URL authorization. File authorization determines which access control lists are consulted based on both the HTTP method being used and the identity making the request. URL authorization is a logical mapping between pieces of the URI namespace and various users or roles


.NET Security
Core Concepts: Authentication
Process of verifying identity by comparing credentials against an authority
Non-authentication = Anonymous user
Basic, Digest, Passport, NTLM, Kerberos and custom or application specific security
11.6

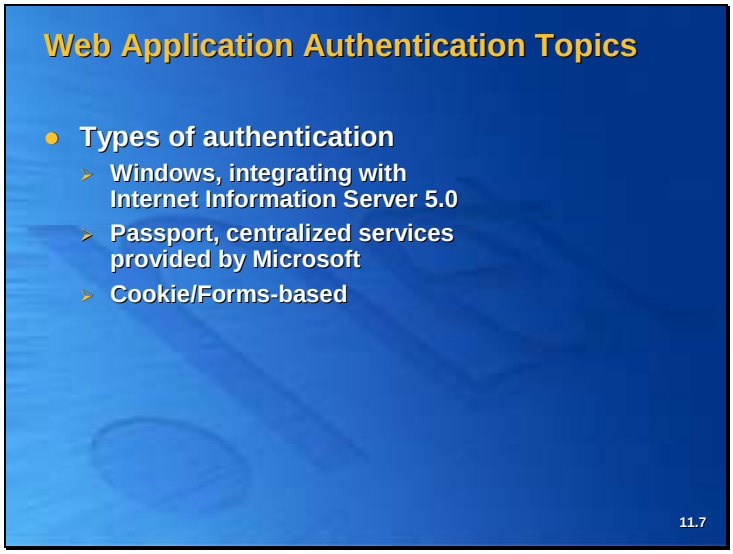
.NET Security

Core Concepts: Authentication

Authentication is the process of discovering and verifying the identity of a principal by examining the user's credentials and validating those credentials against an authority. The information obtained by the authentication process is usable by your code, which means that once the identity of the principal is discovered, developers can use .NET Framework Role-based Security to determine whether to allow that principal to access your code. If authentication fails, the user is identified as anonymous.

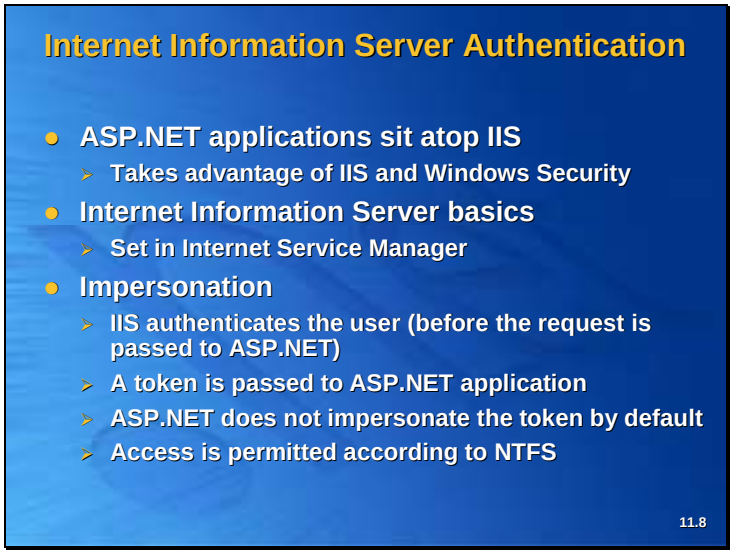
A variety of authentication mechanisms are used in .NET today including many that can be used with .NET Framework Role-based Security. Some of the most commonly used mechanisms are Basic, Digest, Passport, operating system (such as NTLM or Kerberos) or application-defined mechanisms.

 ©2002 Microsoft Corporation



Web Application Authentication Topics

These topics are discussed in the next slides.


Internet Information Server Authentication
ASP.NET applications sit atop IIS
Takes advantage of IIS and Windows Security
Internet Information Server basics
Set in Internet Service Manager
Impersonation
IIS authenticates the user (before the request is passed to ASP.NET)
A token is passed to ASP.NET application
ASP.NET does not impersonate the token by default
Access is permitted according to NTFS
11.8

Internet Information Server Authentication

Web clients communicate with **ASP.NET applications** through IIS, which deciphers and optionally authenticates the request (before ASP.NET ever gets involved). If Allow Anonymous is enabled, no authentication occurs in IIS. IIS also finds the requested resource, such as an ASP.NET application, and if the client is authorized, it returns the appropriate resource. At this point ASP.NET can perform impersonation and/or authentication on the request.

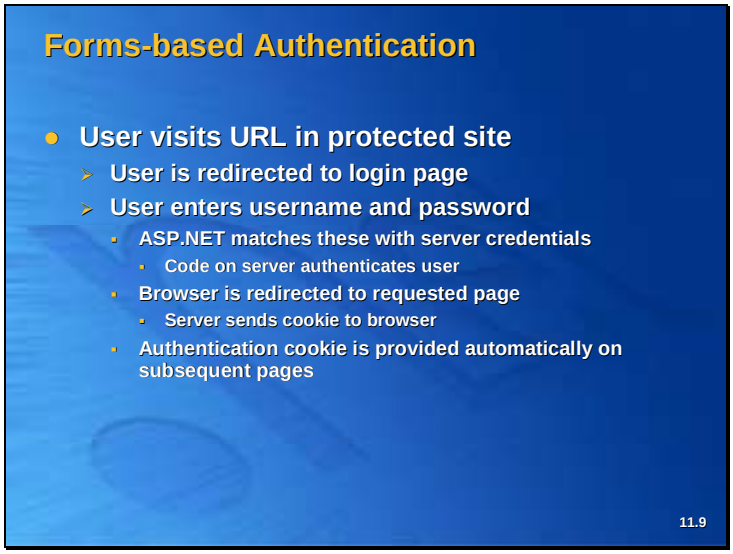
When using **impersonation**, ASP.NET applications can execute with the identity of the client on whose behalf they are operating and in return, developers avoid dealing with authentication and authorization issues in the ASP.NET application code. Instead, you rely on IIS to authenticate the user and either pass an authenticated token to the ASP.NET application or, if unable to authenticate the user, pass an unauthenticated token. In either case, the ASP.NET application impersonates whichever token is received if impersonation is enabled. The ASP.NET application, now impersonating the client, then relies on the settings in the NTFS directories and files to allow access. Be sure to format the server file space as NTFS so access permissions can be set.

For ASP compatibility, impersonation is disabled by default. If impersonation is enabled for a given application, ASP.NET always impersonates the access token that IIS provides to ISAPI extensions. That token can be either an authenticated user token or the token for the anonymous user such as IUSR_MACHINENAME. The impersonation occurs regardless of the type of authentication being used in the application and whether the user is authenticated, in which case, it is the anonymous user token.

Only application code is impersonated while compilation and configuration are read as the process token. If an application is on an UNC share, ASP.NET will impersonate the token provided to IIS to access that share unless a configured account is used. If an explicit configured account is provided, ASP.NET will use that account in preference to the IIS UNC token. Applications that demand per-request impersonation can be configured to impersonate the user making the request.

Impersonation is disabled at the computer level by default and, unless overridden, all the application domains inherit this setting. You can enable impersonation by putting a configuration file in the application root directory.

 ©2002 Microsoft Corporation


Forms-based Authentication
User visits URL in protected site
User is redirected to login page
User enters username and password
ASP.NET matches these with server credentials
Code on server authenticates user
Browser is redirected to requested page
Server sends cookie to browser
Authentication cookie is provided automatically on subsequent pages
11.9

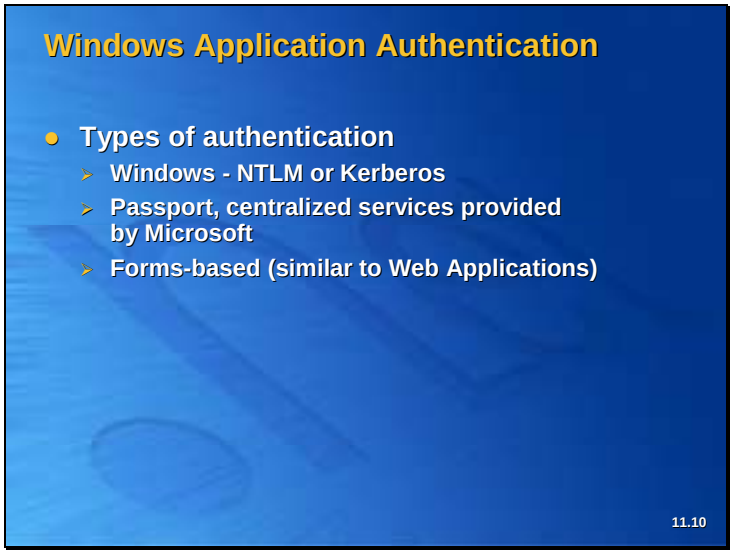
Forms-based Authentication

Build login form with Login.aspx. Obtain the username and password, validate against a trusted data source and call the RedirectFromLoginPage method of FormsAuthentiction class. This will redirect the user to the originally requested URL. After the steps of authentication are complete and the user is identified, .NET handles the remaining process.

Cookie may be temporary (memory only), or persistent (disk-based).

Configuring
Set in Web.Config

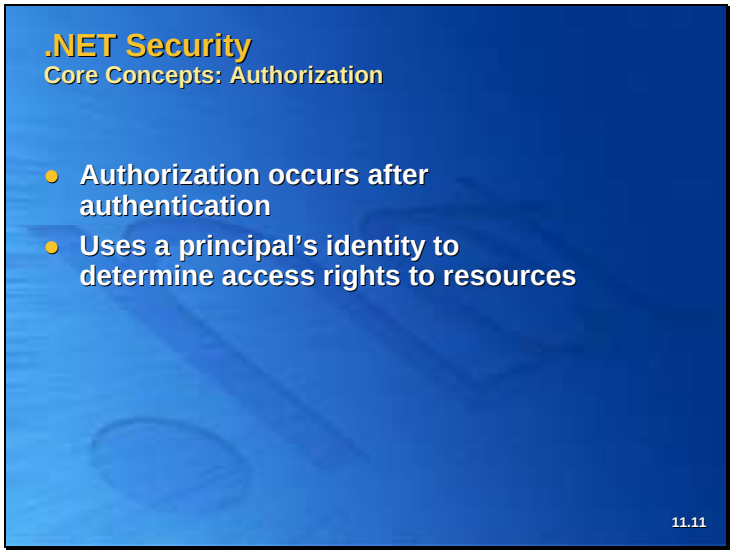
```
<Authentication mode = “Cookie”>
<cookie loginurl = “login.aspx”
      decryptionkey = “autogenerate”
      cookie = “.ASPXCOOKIEAUTH” />
      </Authentication>
```



Windows Application Authentication

A variety of authentication mechanisms are used today, many of which can be used with .NET Framework Role-based Security. Some of the most commonly used mechanisms are BASIC, Digest, Passport, operating system such as NTLM or Kerberos or application-defined mechanisms.

 ©2002 Microsoft Corporation


.NET Security
Core Concepts: Authorization
Authorization occurs after authentication
Uses a principal's identity to determine access rights to resources
11.11

.NET Security

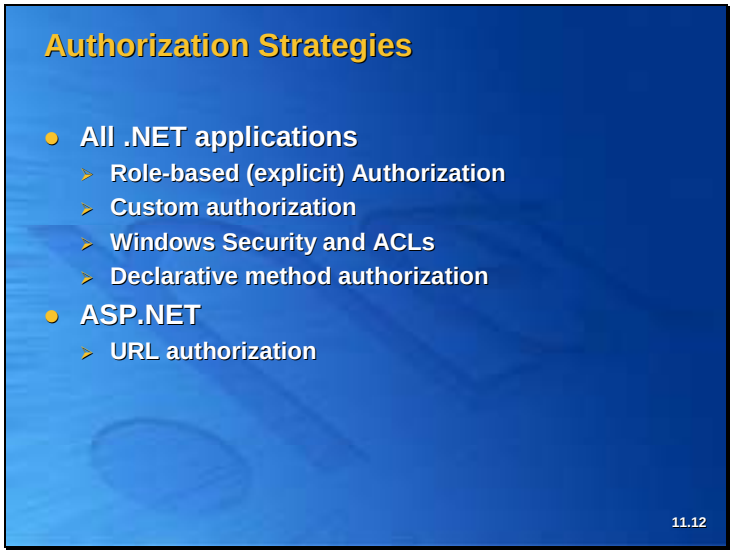
Core Concepts: Authorization

Authorization, which determines whether a principal is allowed to perform a requested action, occurs after authentication and uses information about the principal's identity to determine what resources the principal can access. .NET Framework Role-based Security can be used to implement authorization.

A **principal** represents the identity and role of a user and acts on the user's behalf.

Role-based Security in the .NET Framework supports three kinds of principals:

- Generic principals represent users and roles that exist independent of Windows NT and Windows 2000 users and roles.
- Windows principals represent Windows users and their roles (or their Windows NT and Windows 2000 groups). A Windows principal can impersonate another user, which means the principal can access a resource on a user's behalf while presenting the identity that belongs to that user.
- Custom principals can be defined by an application in any way necessary for that particular application. They can extend the basic notion of the principal's identity and roles.



Authorization Strategies

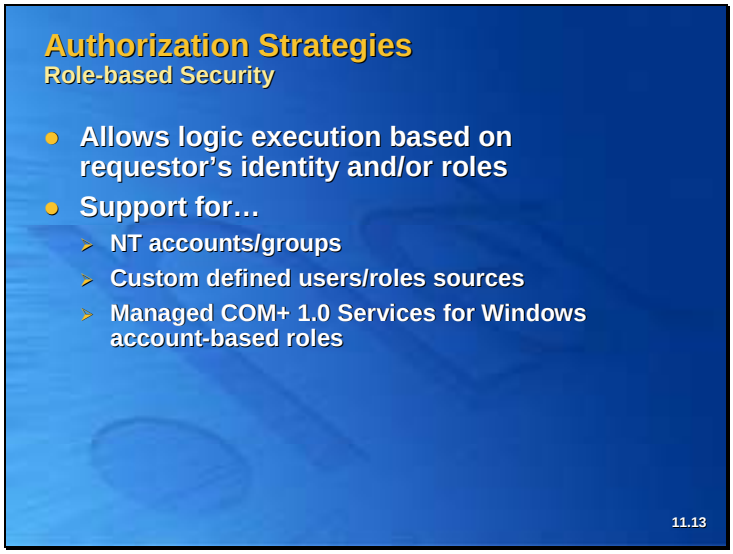
Role-based Authorization - Business applications often provide access to data or resources based on credentials supplied by the user. Typically, applications check the role of a user and provide access to resources based on that role. The Common Language Runtime provides support for Role-based Authorization based on a Windows account or a custom identity.

Authorization occurs after authentication and uses information about the principal's identity and roles to determine what resources the principal can access. You can use .NET Framework Role-based Security to implement authorization.

Role or identity checks may be incorporated into managed code using either **declarative or imperative mechanisms.**

If you request a **URL** containing an ASP.NET application, the request and authentication information are passed to the application.

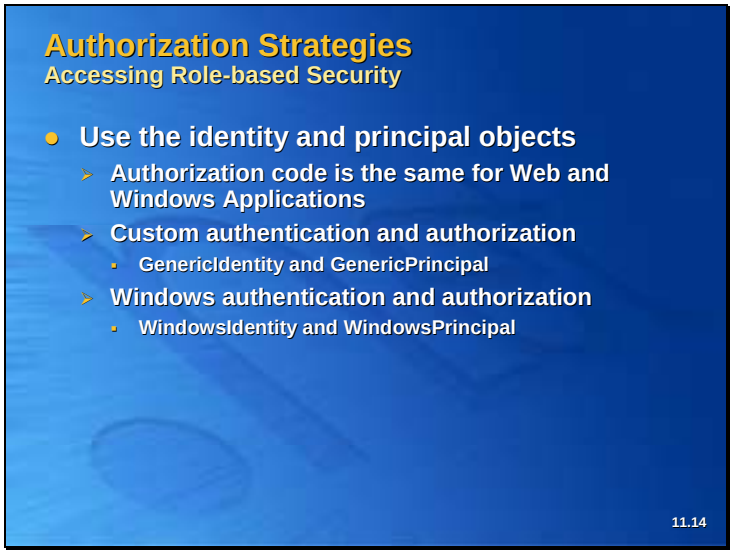
 ©2002 Microsoft Corporation



Authorization Strategies

Role-based Security

Roles are often used in financial or business applications to enforce policy. For example, an application might impose limits on the size of the transaction being processed depending on whether the user making the request is a member of a specified role. Clerks might have authorization to process transactions that are less than a specified threshold, supervisors might have a higher limit and vice-presidents might have a still higher limit or no limit.


Authorization Strategies
Accessing Role-based Security
Use the identity and principal objects
Authorization code is the same for Web and Windows Applications
Custom authentication and authorization
GenericIdentity and GenericPrincipal
Windows authentication and authorization
WindowsIdentity and WindowsPrincipal
11.14

Authorization Strategies

Accessing Role-based Security

The .NET Framework Role-based Security supports authorization by making information about the principal, which is constructed from an associated identity, available to the current thread. The identity, along with the principal it helps to define, can be based on a Windows account or be a custom identity unrelated to a Windows account.

 ©2002 Microsoft Corporation

Authorization Strategies

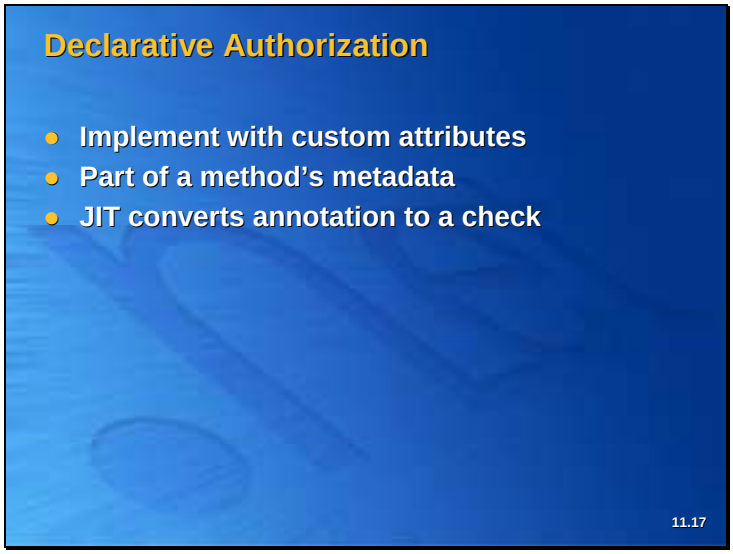
Demo

The purpose of this demo is to illustrate how to access Windows 2000/NT Security information and perform Role-based Security.

Custom Web Authorization
Handle AuthorizeRequest event
Application level (global.asax)
Http module (implement IHttpModule)
Scenarios:
Implement per-request billing system
Restrict access based on the time of day or other custom parameters
Restrict access based on dynamic behaviors (e.g., implement a per-day access limit)
11.16

Custom Web Authorization

 ©2002 Microsoft Corporation



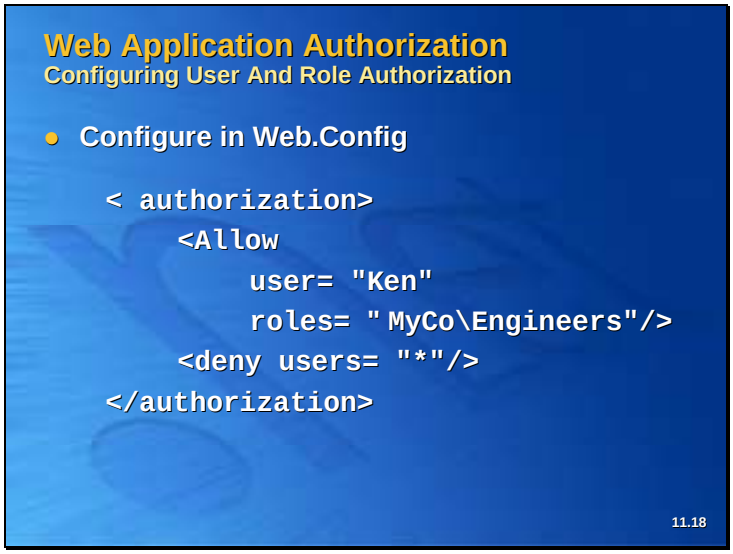
Declarative Authorization

With **Declarative Role-based Security**, developers declare roles — traditionally by using either the Component Services administrative tool or the Administrative SDK functions — and assign them to application resources. Where and how the security is set will determine where an application's boundaries are drawn.

You can assign a role to the entire application, a particular component, a particular interface in a component or a particular method on an interface. Role assignments are inherited through the natural chain of inclusion so if you assign a role to a component, it is implicitly assigned to every interface and method exposed by that component.

With the availability of method-level role assignments, you can secure components and interfaces that were designed without security in mind. If the methods themselves are insecurable with declarative role assignments, you might need to do programmatic role checking. Keep security in mind when deciding how to factor business functionality through methods otherwise or you could be forced to add in security-related code at the last minute.

This is simply a sliver of the Declarative Security feature area. It is presented here because this topic targets Web development, but you can declaratively demand any identity permission (i.e. string name or pub cert) or code access permission.



Web Application Authorization

Configuring User and Role Authorization

The example in this slide allows the user “Ken” and any user in the “Engineers” role to access files in this application while all other users are denied access because of the <deny> tag.

You can map either user accounts or roles to Windows users if you are using IIS Authentication. When doing so, simply set a domain or server name prefix to the username or role. For instance, if the “Engineers” role maps to the “Engineers” group in the MyCo domain, the authorization will appear like this:

```
< authorization><Allow  user= "Ken"
      roles= “MyCo\Engineers"/>
<deny users= "*"/></authorization>
Forcing users to login
? represents anonymous users
< authorization>
      <deny users= “?"/>
</authorization>
```

Example: Allow “Admins” or “WebServiceUsers” and deny all others

```
<!--  * represents all users,
      ? represents anonymous users -->
<authorization>
  <allow Roles= "Admins" />
  <allow Roles= "WebServiceUsers"/>
  <deny users= "*" />
</authorization>
```

 ©2002 Microsoft Corporation

Web Application Authorization

Demo

The purpose of this demo is to illustrate URL Authorization.

Summary

Types of authentication

- Windows, integrating with Internet Information Server 5.0
- Passport, centralized services provided by Microsoft
- Cookie/Forms-based
- Role-based Security support

Types of authorization:

All .NET applications

- Role-based (explicit) Authorization
- Custom Authorization
- Windows Security and ACLs
- Declarative Method Authorization

ASP.NET

- URL authorization

 ©2002 Microsoft Corporation

Resources

Microsoft .NET Framework Security Overview:
http://msdn.microsoft.com/vstudio/nextgen/technology/frameworksec.asp

 ©2002 Microsoft Corporation